Computer Engineering and Intelligent Systems www.iiste.org
ISSN 2222-1719 (Paper) ISSN 2222-2863 (Online)
Vol.4, No.2, 2013

Intelligent Application of Partial Repair for Handling Inconsistency among Database

Abdollah Yousefzadeh* Dr. Hrudya Ku. Tripathy
School of Computing & Technology, Asia Pacific University of Technology & Innovation, Malaysia
* E-mail of the corresponding author: 1

Abstract

Handling inconsistencies among standalone and integrated databases has been an important issue for database's administrators for decades where nowadays databases are huge and not only different types of inconsistencies are ubiquitous in them but also application of any repair might induce new violations to integrity constraints. Resolved data may harm rest of database that leads to a costly process for repair of inconsistent data while after any resolution of data, database should be checked whether any new violation has emerged or not. Introducing partial repair through an approach to measure the tendency that a resolved portion of data incurs new violation would help any repair algorithm to isolate a selection of problematic data (not all), resolve it and save the database from being hurt during repair process. Partial repair keeps the rest of data from being affected that eliminates concerns over application of repair. Partial repair may not handle entire inconsistencies among databases but it represents a repair that would have minimum harm to rest of data along with consideration of cost which makes it valuable.

Keywords: data quality, repair, inconsistency, dependency

1. Introduction

Information is one of the most valuable asset for businesses and organizations (Gordon, 2007) and quality of data in terms of its consistency, completeness and accuracy has been always noticed while data quality plays a decisive role in result of any further process on data, range from making report, data mining and applying complex queries in data warehouses to Knowledge Discovery of Databases as KDD. Therefore detection and correction of poor data has become a step and phase within database management process (Tan, 2006). Once real world data tends to contain inconsistencies (Han, 2006), many works have been done (Bohannon et al., 2005, Cong et al., 2007, Decker, 2011, Fan, 2008, Garc\ et al., 2010, Lian et al., 2011) in order to improve the quality of data through handling the inconsistencies within databases by representing different types of repair. Despite handling inconsistencies can be met through three distinct approaches including repair algorithms, consistent query answering and condensed representation of all repairs, we concentrate on repair algorithms in this research.

Challenges during previous repair processes (Bohannon et al., 2005, Cong et al., 2007, Decker, 2011, Fan, 2008, Garc\ et al., 2010, Lian et al., 2011) can be described as, (1) finding an efficient repair is intractable (Cong et al., 2007) and also (2) repair process is expected to check entire database which is never satisfied completely (Decker, 2011) while it requires time and also enough processing power, and finally (3) repair process has to ensure that application of resolved data, proposed by specific algorithm preserves the integrity constraints (Bohannon et al., 2005) and doesn't violate them when administrators always have been concerned with checking an update whether it preserves integrity constraints or not, therefore repair process is willing to comply with following definition (Decker, 2011) shown below:

$$\mu(D^u, IC) \leq \mu(D, IC) \qquad (1)$$

Here in equation 1, database D along with a set of integrity constraints IC and D^u as application of an update on D, represents μ as an inconsistency metric. Equation 1 shows the violation of application an update should be equals or less than previous state when cost of a repair $cost(Rep)$ must be acceptable but not minimum.

As mentioned by (Redman, 1998) the data error rate, between 1%-5% can be tolerated and would not be harmful which indicates that the application of repair on database ($Rep \oplus D$) may not necessarily lead to 100% consistent database then some researchers provide D_{opt} as the optimum repair where the acceptable difference between D_{opt} and Rep as must be within a predefined boundary ϵ available in equation 2 (Lian et al., 2011).

$$\frac{diff(Repr, D_{opt})}{D_{opt}} \tag{2}$$

Example 1: Illustration of relation schemas $customer$ denoted as $cust$ and $hotel$ has been provided below where in order to capture more inconsistencies; integrity of relations relies on functional dependencies 1 to 4 and conditional functional dependencies once user tends to insert following tupe as t_5(id, name, phone, city, state, zip)=(1, James Hanks, 555-1212, CH, WY, 8255) to the database.
,).

$$cust(id, name, phone, city, state, zip)$$

Table 1- Customer information

tuple	id	name	phone	city	state	zip
t1	1	Alice Watson	555-1212	Mi	FL	3234
t2	3	Tom Jones	444-2222	LA	CA	2355
t3	4	George Ford	444-2222	LA	CA	4688
t4	5	Erik Tan	333-5454	DET	MI	4688

$FD_1: cust[id] \rightarrow cust[NA, PH, CT, ST, zip]$
$FD_2: cust[PH] \rightarrow cust[CT, ST, zip]$
$FD_3: cust[NA, CT, zip] \rightarrow cust[PH]$
$\phi_1: ([zip] \rightarrow [CT, ST], T_1)$

CT	ST	zip
MI	FL	3323
CH	WY	8323

T_1

$$hotel(id, name, star, phone, city, state, zip)$$

For simplifying FDs some abbreviations have been used which are available below:
$NA \rightarrow name$, $ST \rightarrow state$, $PH \rightarrow phone$, $SR \rightarrow star$, $CT \rightarrow city$

Functional dependencies are defined as; each FD denoted by $R[X] \rightarrow R[Y]$ shows X *and* Y as subsets of attributes within relation R and if there is $t_1, t_2 \in R$ which $D(t_1,X) = D(t_2,X)$ then $D(t_1,Y) = D(t_2,Y)$
Above functional dependencies for relation $cust$ are available. Also table 1 illustrates an instance relation by providing 5 tuples $t_1 - t_5$.

Conditional functional dependency: a CFD is ϕ on relation R is illustrated as $(R: X \rightarrow Y, t_p)$ where (1) both X and Y are subsets of $attr(R)$, (2) $X \rightarrow Y$ is a typical functional dependencies and finally t_p is a pattern tableau contains all attributes of X and Y and for each $t_p \in T_p$, $t_p[A]$ is either a constant value in $dom(A)$ or is unnamed variable'-'. CFDs are utilized because they are able to capture more inconsistencies than functional dependencies (Cong et al., 2007) which also have been provided next to table 1.

Acceptable data errors rate and also challenge of introduction of new violation during repair have led to situation that conveys a partial repair approach that maybe suitable and worth studying over total repair in terms of cost and result.

Contributions: Having a partial repair approach represents a situation in which inconsistency is handled, (1) through introducing the term dependency level (*dl*) which measures the probability that a portion of data might induce new violation then (2) the partial repair is iteratively able to start with those portions of data which have the minimum correlation to inducing new violations and also ignores repairs of portions which may induce serious violations to D. Partial repair approach is illustrated in figure 1.

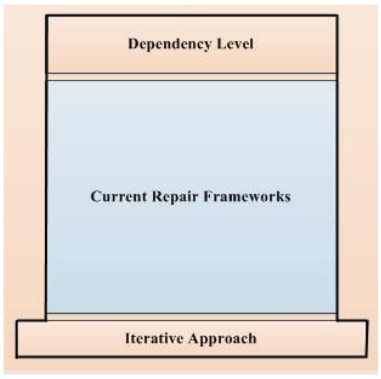

Figure 1. Partial repair framework

In section 2, dependency level will be introduced and in section 3 the contribution of dependency level and partial repair will be demonstrated. Section 4 contains evaluation of partial repair in terms of cost.

2. Dependency level

Having sufficient information about the possible induction of violation needs a comprehensive measurement of the level of tendency that each value $t_i[A]$ where $A \in attr(R)$ has to incur violation after a repair or update which will be called dependency level (or *dl*).

Definition 1: consider a portion p_i containing a group of pairs (t_i, A), for all $t_i[A]$ where $A \in attr(R)$ in p_i dependency level is measured as the summation of total number of FDs, CFDs and INDs in which each $t_i[A]$ in p_i is engaged as $IC_i \subseteq IC$.

$$dl(p_i) = \sum_{i=1}^{n} dl(t_i, A) \qquad (3)$$

In equation 3, n shows the number of pairs inside the portion p_i and IC_i contains either a group of FDs, CFDs and INDs or a set of integrity constraints which pairs in p_i are referred to.

The less the amount of dl of a specific portion of data is, there is more confidence and guarantee that less FDs, CFDs and INDs (or integrity constraints) might be violated through any repair or update during either a repair process or daily transaction of a database. Inclusion dependencies (INDs) are defined in (Bohannon et al., 2005).

The definition dl is constituted of two distinct terms, called inner and outer dependency level. Where inner dependency level indicates the level of dependency that a portion of pairs (t_i, A) are bounded together in terms of shared IC and outer dpedency level reveals the level of dependency that a portion of pairs (t_i, A) have with non-shared integrity constraints.

Definition 2: Inner dependency level (figure 2) refers to; the level in which pairs (t_i, A) in p_i is dependent on shared integrity constraint to IC_i.

Definition 3: Outer dependency level ((figure 2)) refers to; the level in which pairs (t_i, A) in p_i is dependent on non-shared integrity constraint to IC_i.

$$dl(p_i) = dl_{inner}(p_i) + dl_{outer}(p_i) \tag{4}$$

As shown in figure 2, the more $dl_{inner}(p_i)$ is the less violation to non-shared integrity constraints (or group of FDs, CFDS and INDs) will be induced during any update or repair on p_i. And also the less is the $dl_{outer}(p_i)$ there is less bound between values inside the portion and non-shared integrity constraints which makes any further update and repair more safer and harmless.

In figure 2 A is a set consists of arrows when each arrow represents two entities; (1) left arrow is a pair (t_i, A) and (2) right arrow is an integrity constraints (or an element in a group of FDs, CFDs or INDs) which pair is in connection with it

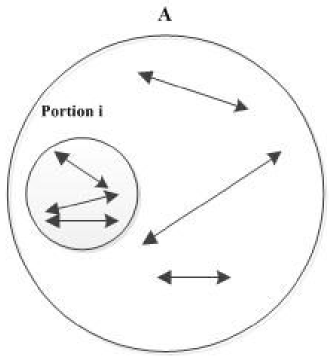

Figure 2. Demonstration of integrity constraints inside database *A as inner* and *outer* dependencies

Therefore in order to keep track of dependency level and also to help any further repair algorithms to benefit from the measurement of dependency level, a new data structure has been proposed which is able to point a group of pairs (t_i, A) to specific amount of dependency level also known as:

$$dl \leftarrow Portion(p_i) \ and \ p_i \leftarrow Portion(dl)$$

Example 2: portion 1 and 2 are available to be repaired which consists of pairs as follows: $p_1 = ((t_1, id), (t_1, CT), (t_4, zip))$ and $p_2 = ((t_5, id), (t_1, PH), (t_3, PH))$ where dl for each portion is:

$$dl(p_1) = 1 + 4 + 4 \ and$$

$$dl(p_2) = 1 + 3 + 3$$

3. Partial repair

Partial repair frame, containing dependency level and current repair mechanisms through an iterative approach will provide the opportunity to handle inconsistency in new and beneficial approach. In an brief explanation, partial repair approach first measure the dependency level of different portions of inconsistent data to order them in an descending approach and calculate and assign efficiency E for each portion, third according to the minimum efficiency that user has defined the repair will be started.

Definition 4: Equation 6 clams the efficiency E of $dl(p_i)$ measured by considering the minimum and maximum of dependency level where $|p_i|$ in equation 5 defines the number of pairs inside the portion and $|IC|$ defines the number of integrity constraints available (or number of FDs, CFDs and INDs).

$$0 \leq dl(p_i) \leq |p_i| \times |IC| \tag{5}$$

And then the efficiency *E* is measured accordingly:

$$E(p_i) = (1 \quad \frac{dl(p_i)}{|p_i| \times |IC|}) \tag{6}$$

Partial repair mechanism can be shown through following procedure below in figure 3.

```
Procedure Partial-Repair (D,$IC$)
Input database D and either a set of integrity constraints or FDs, CFDs and
INDs
Output: Unsolved data
Begin
bestcost←1
Efficiancy($p_i$)←calculate-Efficiency(Portion($p_i$))
While (Exists (Min-Efficiency≤ Efficiancy($p_i$)))
cost($Repr_j$)←(1- Efficiancy($p_i$))
if(cost($Repr_j$ )≤bestcost)then
bestcost←cost($Repr_j$)
UNRESOLVED(i)←portion($dl_i$)
if(cost($Repr_j$)≤bestcost)then
Return(UNSOLVED(i))
END
```

Figure 3. Partial repair algorithm

After dependency levels for all portions have been identified, the efficiency will be measured and will be stored in $Efficiency(p_i)$. if there is any portion of inconsistent data which its efficiency is not less than minimum user defined efficiency, then the $UNSOLVED$ data structure can push portions of inconsistent data to any repair approach. This procedure can be utilized in any repair mechanism as a header which adds new features to the repair algorithm.

4. Cost of Partial Repair

Partial repair may not handle entire inconsistencies among databases but it benefits administrators by providing approach which induces less violation to the rest of the database. In order to evaluate the cost of partial repair, it must be noticed that partial repair is a frame which includes any repair approach and adds more features to it, therefore depending on the type of repair administrator chooses, the cost changes but evaluating the cost of Partial-Repair procedure as a partial repair approach to the type of repair is:

$$cost(Repr_i) = \sum_{j=1}^{m} (\frac{dl(p_j)}{|p_j| \times |IC|}) + cost(D', D) \tag{7}$$

According to the equation 7 cost of resolving data is $cost(D',D)$ where m indicates the number of portions which have returned from the Partial-Repair procedure. Also the time that more than one group of portions has been returned by defining different minimum of efficiency at each time, cost will be calculated as below:

$$cost(Repr) = \sum_{j=1}^{m}\sum_{i=1}^{n}\left(\frac{1}{g} \times \frac{dl(p_j)}{|p_j| \times |IC|}\right) + cost(D', D) \tag{8}$$

In equation 8 where m indicates the number of different groups of portions of inconsistent data, n defines the number of portions in each group and if more than one portion are combined together g defines their number. In example 1, the cost of application portions 1 and 2 together demonstrated in equation 9 where $m = 1,\ n = 2\ and\ g = 2$ and for application of portion 1 individually equals, (3/4) + *cost(D',D)* and for portion 2 equals (7/12)+*cost(D',D)* where $m = 2,\ n = 1\ and\ g = 1$.

$$\left(\frac{9}{24}\right) + \left(\frac{7}{24}\right) + cost(D',D) = \left(\frac{2}{3}\right) + cost(D',D) \tag{9}$$

$cost(Repr_i)$ Is used in procedure Partial-Repair in order to keep track of performance of the repair and tries to help the procedure to produce a repair (from selected portions) which leads to an acceptable cost. Cost $cost(Repr_j)$ is a complement to the efficiency of the portions when it covers the whole process and not only a specific portion.

Example 3: for handling inconsistencies among table $cust$ (table 1), three portions of data are available below and information about dependency level and cost are available in table 2.

$$p_1 = ((t_1, id), (t_1, CT), (t_4, zip))$$

$$p_2 = ((t_5, id), (t_1, PH), (t_3, PH))$$

$$p_3 = ((t_1, id), (t_4, zip), (t_1, PH))$$

Table 1. Measurement of *dl*, *E* and *cost*

Portion	dl	$E(p_j)$	$cost(Repr_j)$	$cost(Repr)$	Priority To repair
p_1	9	0.25	0.75	-	3
p_2	7	0.42	0.58	-	1
p_3	8	0.34	0.66	-	2
p_2, p_3	-	0.38	-	0.62	-

As it is clear from table 2, the efficiency of portion 2 is the best and if the minimum efficiency is set to 0.3, portions 2 and 3 are allowed to be repaired, then cost of portion 2 and 3 as they both be inserted to

$UNSOLVED$ data structure is 0.62 which is also lower than handling only portion 3 individually while more data is going to be resolved.

5. Conclusion

Providing the opportunity to administrators to conduct repairs which result in less violation to the database will have a great impact on efforts to gain quality data within our databases. Through partial repair approach, inconsistent data will be considered as different portions when the determination of which portions to be resolved is based on the measurement of dependency level of each portion and the efficiency it has, therefore inconsistent data will be step by step (portion by portion) resolved until the required efficiency is met. Even though partial repair takes resolving of data into account, it doesn't necessarily resolved entire inconsistencies.

References

Gordon, K., 2007. *Principles of Data Management Facilitating Information Sharing*. Swindon: The Brithish Computer Society.

Tan, P.-N., Steinbach, M. & Kumar, V., 2006. *Introduction to Data Mining*. Pearson International ed. Boston: Pearson Education, Inc.

Han, J. & Kamber, M., 2006. *Data Mining Concepts and Techniques*. 2nd ed. San Francisco: Diane Cerra.

Gao Cong, Wenfei Fan, Floris Geerts, Xibei Jia, and Shuai Ma. 2007. Improving data quality: consistency and accuracy. In Proceedings of the 33rd international conference on Very large data bases (VLDB '07). VLDB Endowment 315-326.

Decker, H., 2011. *Flexible Repairs for Improving the Integrity of Databases*. In International Workshop on Database and Expert Systems Applications. Valencia,Spain, 2011. IEEE.

Philip Bohannon, Wenfei Fan, Michael Flaster, and Rajeev Rastogi. 2005. *A cost-based model and effective heuristic for repairing constraints by value modification*. In Proceedings of the 2005 ACM SIGMOD international conference on Management of data (SIGMOD '05). ACM, New York, NY, USA, 143-154.

Thomas C. Redman. 1998. *The impact of poor data quality on the typical enterprise*. Commun. ACM 41, 2 (February 1998), 79-82. HYPERLINK "http://doi.acm.org/10.1145/269012.269025"

Xiang Lian, Yincheng Lin, and Lei Chen. 2011. *Cost-efficient repair in inconsistent probabilistic databases*. ACM, New York, NY, USA, 1731-1736.

Javier Garcia-Garcia and Carlos Ordonez. 2010. *Repairing OLAP queries in databases with referential integrity errors*. ACM, New York, NY, USA, 61-66.

Wenfei Fan. 2008. *Dependencies revisited for improving data quality*. ACM, New York, NY, USA, 159-170.

Ali A. Alwan, Hamidah Ibrahim, and Nur Izura Udzir. 2009. *Ranking and selecting integrity tests in a distributed database*. ACM, New York, NY, USA, 185-192.

Tennyson X. Chen, Martin D. Meyer, and Nanthini Ganapathi. 2009. *Implementation considerations for improving data integrity in normalized relational databases*. ACM, New York, NY, USA, , Article 3 , 6 pages.

Lijun Ma; , "*The integrity rules and constraints of database*," Electronic and Mechanical Engineering and Information Technology (EMEIT), 2011 International Conference on , vol.7, no., pp.3747-3749, 12-14 Aug. 2011

Getta, J.R.; , "*Efficient Processing of Databases with Inconsistent Information*," Computational Intelligence and Software Engineering (CiSE), 2010 International Conference on , vol., no., pp.1-5, 10-12 Dec. 2010

Ibrahim, H.; Alwan, A.A.; Udzir, N.I.; , "*Checking Integrity Constraints with Various Types of Integrity Tests for Distributed Databases*," Parallel and Distributed Computing, Applications and Technologies, 2007. PDCAT '07. Eighth International Conference on , vol., no., pp.151-152, 3-6 Dec. 2007